for Harry

```
                        TITLE  'COLLEEN FLOATING POINT ROUTINES BY C SHAW'
                        ;
                        ;       MORE ACCURATE VERSIONS OF THE FOLLOWING SHEPARDSON ROUTINES
                        ;
                        ;       EXP, EXP10, LOG, LOG10, SIN, COS, ATAN, SQR AND POWER
                        ;
                        ;       THESE ROUTINES WERE TAKEN FROM THE CALCULATOR CARTRIDGE AND MODIFIED
                        ;       MANY OTHER MATH FUNCTIONS SUCH AS TAN, ARCSIN AND ARCCOS ARE ALSO
                        ;       INCLUDED IN THAT CARTRIDGE.
                        ;
009B                    CR      =       $9B             ;ATASCII CARRIAGE RETURN

0005                    GETREC  =       5               ;GET RECORD
0009                    PUTREC  =       9               ;PUT RECORD
0342                    ICCOM   =       $342
0344                    ICBAL   =       $344
0348                    ICBLL   =       $348
E456                    CIOV    =       $E456

                        ;
                                                        FLOATING POINT SUBROUTINES
                        ;
0006                    FPREC   =       6               ;FLOATING PT PRECISION (# OF BYTES)
                        ;                               IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARRY SET => ERROR
D800                    AFP     =       $D800           ;ASCII->FLOATING POINT (FP)
                        ;                               INBUFF+CIX -> FRO, CIX, CARRY
D8E6                    FASC    =       $D8E6           ;FP -> ASCII        FRO-> LBUFF  (INBUFF)
D9AA                    IFP     =       $D9AA           ;INTEGER -> FP
                        ;                               0-$FFFF (LSB,MSB) IN FRO,FRO+1->FRO
D9D2                    FPI     =       $D9D2           ;FP -> INTEGER      FRO -> FRO,FRO+1,  CARRY
DA60                    FSUB    =       $DA60           ;FRO <- FRO - FR1   ,CARRY
DA66                    FADD    =       $DA66           ;FRO <- FRO + FR1   ,CARRY
DADB                    FMUL    =       $DADB           ;FRO <- FRO * FR1   ,CARRY
DB28                    FDIV    =       $DB28           ;FRO <- FRO / FR1   ,CARRY
DD89                    FLDOR   =       $DD89           ;FLOATING LOAD REG0     FRO <- (X,Y)
DD98                    FLD1R   =       $DD98           ;     "      "   REG1     FR1 <- (X,Y)
DDA7                    FSTOR   =       $DDA7           ;FLOATING STORE REG0 (X,Y) <- FRO
DDB6                    FMOVE   =       $DDB6           ;FR1 <- FRO
DD40                    PLYEVL  =       $DD40           ;FRO <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I)      CARRY
                        ;                               INPUT:  (X,Y) = A(N),A(N-1)...A(0)  -> PLYARG
                        ;                                       ACC   = # OF COEFFICIENTS = DEGREE+1
                        ;                                       FRO   = Z
DDC0                    EXP     =       $DDC0           ;FRO <- E**FRO = EXP10(FRO * LOG10(E))  CARRY
DDCC                    EXP10   =       $DDCC           ;FRO <- 10**FRO  CARRY
DECD                    LOG     =       $DECD           ;FRO <- LN(FRO) = LOG10(FRO)/LOG10(E)   CARRY
DED1                    LOG10   =       $DED1           ;FRO <- LOG10 (FRO)     CARRY

                        ;                               THE FOLLOWING ARE IN BASIC CARTRIDGE.
                        ;SIN    =       $BD81           ;FRO <- SIN(FRO)  DEGFLG=0 =>RADS,  6=>DEG    CARRY
                        ;COS    =       $BD73           ;FRO <- COS(FRO)    CARRY
                        ;ATAN   =       $BE43           ;FRO <- ATAN(FRO)   CARRY
                        ;SQR    =       $BEB1           ;FRO <- SQUAREROOT(FRO)   CARRY

                        ;FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF F.P. ROUTINES ARE CALLED)
                                *=$D4
00D4                    FR0     *=*+FPREC       ;FP REG0
00DA                    FRE     *=*+FPREC
00E0                    FR1     *=*+FPREC       ;FP REG1
00E6                    FR2     *=*+FPREC
```

```
                        ;FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF F.P. ROUTINES ARE CALLED)
                                *=$04
00D4                    FR0     *=*+FPREC       ;FP REG0
00DA                    FR1     *=*+FPREC
00E0                    FR2     *=*+FPREC       ;FP REG1
00E6                    FR3     *=*+FPREC

COLLEEN FLOATING POINT ROUTINES BY C SHAW

00EC                    FRX     *=*+1           ;FP SPARE
00ED                    EEXP    *=*+1           ;VALUE OF E
00EE                    NSIGN   *=*+1           ;SIGN OF #
00EF                    ESIGN   *=*+1           ;SIGN OF EXPONENT
00F0                    FCHRFLG *=*+1           ;1ST CHAR FLAG
00F1                    DIGRT   *=*+1           ;# OF DIGITS RIGHT OF DECIMAL
00F2                    CIX     *=*+1           ;CURRENT INPUT INDEX
00F3                    INBUFF  *=*+2           ;POINTS TO USER'S LINE INPUT BUFFER
00F5                    ZTEMP1  *=*+2
00F7                    ZTEMP4  *=*+2
00F9                    ZTEMP3  *=*+2
00FB                    RADFLG  *=*+1           ;0=RADIANS, 6=DEGREES
00FC                    FLPTR   *=*+2           ;POINTS TO USER'S FLOATING PT NUMBER
00FE                    FPTR2   *=*+2

                        ;FLOATING PT ROUTINES' NON-ZERO PAGE RAM (NEEDED ONLY IF F.P. ROUTINES CALLED)
                                *=$57E
057E                    LBPR1   *=*+1           ;LBUFF PREFIX 1
057F                    LBPR2   *=*+1           ;LBUFF PREFIX 2
0580                    LBUFF   *=*+128         ;LINE BUFFER
05E0                    PLYARG  =       LBUFF+$60       ;POLYNOMIAL ARGUMENTS
05E6                    FPSCR   =       PLYARG+FPREC
05EC                    FPSCR1  =       FPSCR+FPREC
05E6                    FSCR    =       FPSCR
05EC                    FSCR1   =       FPSCR1
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
COLLEEN FLOATING POINT ROUTINES BY C SHAW

                                        FP PACKAGE EQUATES FOR SIN, COS, ATAN, AND SQR ROUTINES ETC

0008            NATCF   =       $8      ;NUMBER OF ATAN COEFFICIENTS FOR POLYNOMIAL EVALUATION
0006            NSCF    =       6       ;NUMBER OF SIN COEFFICIENTS

D905            FASC2   =       $D905   ;AFTER FASC (FINISH FP TO ASCII CONVERSION)
D920            XEFORM  =       $D920   ;!EFORM   PROCESS E FORMAT    FOR FP -> ASCII CONVERSION
D928            XEFRM2  =       $D928   ;AFTER XEFORM (FINISH CONVERSION)
DA44            ZFRO    =       $DA44   ;FRO <- 0
DA46            ZF1     =       $DA46   ;CLEAR 6 BYTES STARTING AT 0,X
DA51            INTLBF  =       $DA51   ;INIT LBUFF INTO INBUFF    FOR FP -> ASCII CONVERSION
DC00            NORM    =       $DC00   ;NORMALIZE FLOATING POINT NUMBER - USED BY STRUNC ONLY
DC70            XCVFRO  =       $DC70   ;!CVFRO         FP TO 10 ASCII DIGITS IN LBUFF
DE03            EXP1    =       $DE03   ;MIDDLE OF EXP10 WHERE PLYEVL IS CALLED
DE12            EXP11   =       $DE12   ;AFTER PLYEVL IN EXP10
DE89            LOG10E  =       $DE89   ;LOGTEN(E) = .4342944819
DE95            XFORM   =       $DE95   ;FRO <- (FRO-(X,Y)) / (FRO+(X,Y))
DF6C            FHALF   =       $DF6C   ;FLOATING POINT CONSTANT .5
DFAE            ATCOEF  =       $DFAE   ;ATAN COEFFICIENTS
DFEA            FP9S    =       $DFEA   ;FLOATING POINT CONSTANT .9999999999 (ALMOST 1)
DFF0            PIOV4   =       $DFF0   ;FLOATING POINT CONSTANT PI/4 = .7853981634


                                        VARIABLES
                        *=$480
0480            QUADFLG *=*+1           ;SIN QUADRANT FLAG
0481            INTFLG  *=*+1           ;FLAG FOR POWER ROUTINE
0482            FTEMP   *=*+6           ;TEMPORARY FLOATING POINT REGISTER FOR POWER ROUTINE
```

```
                                *=$A000                 ;ARBITRARY STARTING POINT
                        ;
                        ;       TEST PROGRAM
                        ;
A000                    START
A000 20 4C A0                   JSR     GETNUM
A003 20 B6 DD                   JSR     FMOVE
A006 20 4C A0                   JSR     GETNUM          ;GET 2ND NUMBER FROM E: -- OMIT IF ONLY ONE ARGUMENT

A009 20 CE A0                   JSR     SPOWER          ;CHANGE TO GET DIFFERENT ROUTINES

A00C 90 0A                      BCC     NOERR
                        ;
                        ;       ERROR -- DISPLAY MESSAGE
                        ;
A00E A9 79                      LDA     #ERRMSG
A010 8D 44 03                   STA     ICBAL
A013 A9 A0                      LDA     #ERRMSG/256
A015 4C 32 A0                   JMP     CONTIN

A018                    NOERR
A018 20 E6 D8                   JSR     FASC            ;FLOATING POINT TO ASCII
                        ;
                        ;       FIND END OF STRING AND CHANGE NEGATIVE # TO POSITIVE AND ADD CARRIAGE RETURN
                        ;
A01B A0 FF                      LDY     #$FF
A01D                    MLOOP
A01D C8                         INY
A01E B1 F3                      LDA     (INBUFF),Y
A020 10 FB                      BPL     MLOOP
A022 29 7F                      AND     #$7F
A024 91 F3                      STA     (INBUFF),Y
A026 C8                         INY
A027 A9 9B                      LDA     #CR
A029 91 F3                      STA     (INBUFF),Y

                        ;
                        ;       DISPLAY RESULT
                        ;
A02B A5 F3                      LDA     INBUFF
A02D 8D 44 03                   STA     ICBAL
A030 A5 F4                      LDA     INBUFF+1
A032                    CONTIN
A032 8D 45 03                   STA     ICBAL+1
A035 A9 09                      LDA     #PUTREC
A037 8D 42 03                   STA     ICCOM
A03A A9 28                      LDA     #40
A03C 8D 48 03                   STA     ICBLL
A03F A9 00                      LDA     #0
A041 8D 49 03                   STA     ICBLL+1
A044 A2 00                      LDX     #0
A046 20 56 E4                   JSR     CIOV

A049 4C 00 A0                   JMP     START           ;DO IT AGAIN

A04C                    GETNUM                  ;GET ONE NUMBER FROM E: (IOCB #0)
A04C A9 05                      LDA     #GETREC         ;GET RECORD (ENDS IN CR)
A04E 8D 42 03                   STA     ICCOM
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
A04C                    GETNUM                  ;GET ONE NUMBER FROM E: (IOCB #0)
A04C A9 05                      LDA     #GETREC         ;GET RECORD (ENDS IN CR)
A04E 8D 42 03                   STA     ICCOM
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
A051 A9 80                      LDA     #LBUFF
A053 8D 44 03                   STA     ICBAL
A056 A9 05                      LDA     #LBUFF/256
A058 8D 45 03                   STA     ICBAL+1
A05B A9 28                      LDA     #40
A05D 8D 48 03                   STA     ICBLL
A060 A9 00                      LDA     #0
A062 8D 49 03                   STA     ICBLL+1
A065 A2 00                      LDX     #0
A067 20 56 E4                   JSR     CIOV
A06A A9 80                      LDA     #LBUFF
A06C 85 F3                      STA     INBUFF
A06E A9 05                      LDA     #LBUFF/256
A070 85 F4                      STA     INBUFF+1
A072 A9 00                      LDA     #0
A074 85 F2                      STA     CIX
A076 4C 00 D8                   JMP     AFP             ;CALL ASCII TO FLOATING POINT AND RETURN

A079 45 52 52   ERRMSG  .BYTE   "ERROR",CR              ;INDICATES CARRY SET RETURN FROM FP ROUTINE
A07C 4F 52 9B
```

```
                        ;
                        ;       FR0 <- E^FR0
                        ;
                        ;       USES INTEGER FUNCTION LIKE BASIC'S INSTEAD OF JUST IFP, WHICH ROUNDS
                        ;       PROVIDES ACCURACY OF AT LEAST 7 DIGITS (EXCEPT POSSIBLY AT EXTREMA)
                        ;       INSTEAD OF 6.
                        ;
A07F                    SEXPE                           ;E^X (SEE SHEP ATARI BASIC $DDC0 EXP)
A07F A2 89                      LDX     #LOG10E         ;E^X = 10^(X*LOGTEN(E))
A081 A0 DE                      LDY     #LOG10E/256
A083 20 E8 A2                   JSR     LD1MUL          ;FR0 <- FR0*LOG10E
                        ;
                        ;       FR0 <- 10^FR0           (SEE COMMENTS FOR SEXPE)
                        ;
                        ;       RETURNS EXACT POWER OF 10 FOR INTEGERS.
                        ;
A086                    SEXPTE                          ;10^X (SEE SHEP ATARI BASIC $DDCC EXP10)
A086 A9 00                      LDA     #0              ;CLEAR TRANSFORM FLAG
A088 85 F1                      STA     DIGRT           ;XFMFLG
A08A A5 D4                      LDA     FR0
A08C 85 F0                      STA     FCHRFLG         ;SAME AS SGNFLG   REMEMBER ARG SIGN
A08E 20 D2 A2                   JSR     SABSVA          ;TAKE ABSOLUTE VALUE, A<-FR0
A091 38                         SEC
A092 E9 40                      SBC     #$40
A094 30 27                      BMI     SEXP05          ;X<1 SO USE SERIES DIRECTLY (BUT CHECK FOR 0 FIRST)
A096 A2 E6                      LDX     #FPSCR
A098 A0 05                      LDY     #FPSCR/256
A09A 20 A7 DD                   JSR     FSTOR           ;SAVE IN SCRATCH REG
A09D 20 18 A3                   JSR     SINTEG          ;GREATEST INTEGER <= X
A0A0 20 D2 D9                   JSR     FPI             ;MAKE INTEGER
A0A3 B0 27                      BCS     SFERR3          ;RETURN IF ERROR
A0A5 A5 D5                      LDA     FR0+1           ;CHECK MSB
A0A7 D0 23                      BNE     SFERR3          ;SHOULDN'T HAVE ANY -- RETURN IF ERROR
A0A9 A5 D4                      LDA     FR0
A0AB 85 F1                      STA     DIGRT           ;XFMFLG         SAVE MULTIPLIER EXP
A0AD 20 AA D9                   JSR     IFP             ;NOW TURN IT BACK TO FP
A0B0 20 B6 DD                   JSR     FMOVE           ;FR1 <- FR0
A0B3 A2 E6                      LDX     #FPSCR          ;RELOAD FROM TEMP SCRATCH REG
A0B5 A0 05                      LDY     #FPSCR/256
A0B7 20 89 DD                   JSR     FLDOR
A0BA 20 07 A3                   JSR     SFSUB
A0BD                    SEXP05
A0BD A5 D4                      LDA     FR0
A0BF D0 08                      BNE     SEXP10
A0C1 A9 01                      LDA     #1              ;10^0 = 1
A0C3 20 53 A3                   JSR     PSET0
A0C6 4C 12 DE                   JMP     EXP11           ;$DE12  DO 10^X, SKIPPING PLYEVL   LDA XFMFLG
A0C9                    SEXP10
A0C9 4C 03 DE                   JMP     EXP1            ;DO REST OF 10^X
A0CC                    SFERR3
A0CC 38                         SEC
A0CD 60                 INIT    RTS
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
COLLEEN FLOATING POINT ROUTINES BY C SHAW


                        ;       FRO <- FRO ^ FR1 = EEXPTE (FR1 * SLOGTE (FRO))
                        ;
                        ;       USES MORE ACCURATE EEXPTE INSTEAD OF EXP10
                        ;       RETURNS EXACT INTEGER IF BOTH FRO AND FR1 ARE POSITIVE INTEGERS.
                        ;       RETURNS RECIPROCAL OF INTEGER IF BOTH ARE INTEGERS AND FR1 < 0
                        ;       RETURNS CARRY SET IF FRO < 0 OR (FRO = 0 AND FR1 < 0) OR OVERFLOW
                        ;       0 ^ FR1 = 0 IF FR1 > 0
                        ;       0 ^ 0 = 1
A0CE                    SPOWER
A0CE A5 D4                      LDA     FRO             ;FRO = 0?
A0D0 D0 0D                      BNE     SPOW20          ;NO.
A0D2 A9 00                      LDA     #0              ;YES.
A0D4 A6 E0                      LDX     FR1
A0D6 30 78                      BMI     PERR2           ;FR1 < 0        0 ^ -X  => ERROR
A0D8 D0 02                      BNE     SPOW10          ;FR1 > 0        0 ^  X  =  0
A0DA A9 01                      LDA     #1              ;FR1 = 0        0 ^  0  =  1
A0DC                    SPOW10
A0DC 4C 53 A3                   JMP     PSETO
A0DF                    SPOW20
A0DF A5 E0                      LDA     FR1
A0E1 48                         PHA                     ;SAVE FR1'S SIGN
A0E2 29 7F                      AND     #$7F            ;TAKE ABSOLUTE VALUE OF FR1
A0E4 85 E0                      STA     FR1
A0E6 A2 82                      LDX     #FTEMP          ;SAVE FR1 IN FTEMP
A0E8 A0 04                      LDY     #FTEMP/256
A0EA 20 D9 A2                   JSR     FST1R
A0ED 20 B6 DD                   JSR     FMOVE
A0F0 A9 01                      LDA     #1
A0F2 8D 81 04                   STA     INTFLG          ;ASSUME NOT BOTH INTEGERS
A0F5 20 31 A3                   JSR     STRUNC          ;TRUNCATE FRO -- RETURN A=0 AND EQ IF FRO WAS ALREADY AN INTEGER
A0F8 D0 0F                      BNE     SPOW50          ;FRO WAS NOT AN INTEGER
A0FA A2 82                      LDX     #FTEMP          ;LOAD SAVED VALUE INTO FRO
A0FC A0 04                      LDY     #FTEMP/256
A0FE 20 89 DD                   JSR     FLDOR
A101 20 31 A3                   JSR     STRUNC          ;TEST FOR INTEGER
A104 D0 03                      BNE     SPOW50          ;NOT INTEGER
A106 8D 81 04                   STA     INTFLG          ;0 => BOTH INTEGER => RESULT SHOULD BE INTEGER
A109                    SPOW50
A109 A2 E0                      LDX     #FR1
A10B A0 00                      LDY     #FR1/256        ;FRO <- FR1 (MOVE ORIGINAL FRO BACK)
A10D 20 89 DD                   JSR     FLDOR
A110 20 38 A1                   JSR     SLOGTE          ;LOG10(FRO)
A113 B0 3A                      BCS     PERROR          ;ERROR => POP FR1 SIGN AND RETURN
A115 A2 82                      LDX     #FTEMP          ;LOAD FR1 AGAIN
A117 A0 04                      LDY     #FTEMP/256
A119 20 98 DD                   JSR     FLD1R
A11C 20 DB DA                   JSR     FMUL            ;FRO <- FR1 * LOG10(BASE)
A11F B0 2E                      BCS     PERROR          ;RETURN IF ERROR
A121 20 86 AD                   JSR     SEXPTE          ;10 ^ FRO
A124 B0 29                      BCS     PERROR
A126 AD 81 04                   LDA     INTFLG          ;SHOULD RESULT BE INTEGER?
A129 D0 15                      BNE     SPOW80          ;NO.
                        ;                               ;YES -- ROUND TO NEAREST INTEGER
A12B A2 6C                      LDX     #FHALF          ;FR1 <- 0.5
A12D A0 DF                      LDY     #FHALF/256
A12F 20 98 DD                   JSR     FLD1R
A132 A5 D4                      LDA     FRO
A134 10 04                      BPL     SROU10
```

```
A13A A9 BF              LDA     #$3F+$80        ;IF FR0 <0 THEN FR1 <- -0.5
A13C 85 E0              STA     FR1
A13E            SADD1
A13E 20 F7 A2           JSR     SFADD           ;FR0 <- FR0 + FR1  (2-LEVEL RETURN IF ERROR)
A141 20 31 A3           JSR     STRUNC          ;TRUNCATE

A144            SFOMBO
A144 18                 CLC                     ;INDICATE NO ERROR?
A145 68                 PLA                     ;RELOAD FR1'S ORIGINAL SIGN
A146 10 0D              BPL     PRTN            ;DONE IF > 0
A148 20 B6 DD           JSR     FMOVE           ;IF < 0 THEN TAKE RECIPROCAL
A14B A9 01              LDA     #1
A14D 20 53 A3           JSR     PSET0           ;FR0 <- 1
A150 4C 28 DB           JMP     FDIV

A153            FERROR
A153 68                 PLA                     ;DISCARD FR1'S SIGN
A154            FERR2
A154 38                 SEC                     ;INDICATE ERROR
A155            PRTN
A155 60                 RTS
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
COLLEEN FLOATING POINT ROUTINES BY C SHAW


                        ;
                        ;       FRO <- NATURAL LOG (FRO)
                        ;
                        ;       RETURNS CARRY SET IF FRO<=0
                        ;       RETURNS EXACTLY 0 IF FRO = 1
                        ;
A152                    LN
A152 20 5E A1                   JSR     LOGCHK          ;CHECK FOR 0,1 (SPECIAL CASES)
A155 4C CD DE                   JMP     LOG


                        ;
                        ;       FRO <- COMMON LOG (FRO)         (LOG BASE 10)
                        ;
                        ;       SIMILAR TO SLN
                        ;
A158                    LOG10
A158 20 5E A1                   JSR     LOGCHK
A15B 4C D1 DE                   JMP     LOG10

A15E                    LOGCHK                          ;CHECK FOR 0,1
A15E 38                         SEC
A15F A5 D4                      LDA     FRO
A161 F0 13                      BEQ     PULRTN          ;LN(0),LOG(0) => ERROR
A163 30 11                      BMI     PULRTN          ;<0  => ERROR  =>  2-LEVEL RETURN
A165 A2 05                      LDX     #FPREC-1
A167                    LOGCLP
A167 B5 D4                      LDA     FRO,X
A169 DD A9 A3                   CMP     ONE,X
A16C D0 0A                      BNE     RTURN2          ;NOT 1 => OK
A16E CA                         DEX
A16F 10 F6                      BPL     LOGCLP
A171 68                         PLA                     ;SKIP LOGCHK RETURN
A172 68                         PLA
A173 4C 51 A3                   JMP     PCLR0           ;LN(1)=LOGTEN(1)=0
A176                    PULRTN
A176 68                         PLA
A177 68                         PLA
A178 60                 RTURN2  RTS
```

```
                        ; BASIC SINE & COS ROUTINES
                        ;
                        ; TO FIX BUGS OF VERSION 5.9 OF SHEP BASIC
                        ;
                        ; BY DAVE & LARRY -- MODIFIED BY CAROL
                        ; 4-6-79
                        ;
                        ; MOD FUNCTION MAKES ROUTINES MORE ACCURATE FOR ANGLES > 360 DEGREES
                        ;
                        ; COSINE ROUTINE -- ADD 90 OR PI/2 TO FR0 TO DO SIN
A179                    SCOS
A179 20 6F A3                   JSR     SINMOD          ;TAKE ANGLE MOD 2*PI, 360
A17C 20 A0 A3                   JSR     PIOVL           ;SET UP X & Y REGS TO LOAD PI/2 OR 90
A17F 20 9B DD                   JSR     FLD1R           PUT PI/2 OR 90 INTO FR1
A182 20 F7 A2                   JSR     SFADD           FR0=FR0 + PI/2 (OR 90)
                        ;
                        ; SINE ROUTINE
                        ; COMPUTE QUADRANT, GET FRACTION AND DO POLYNOMIAL,
                        ; THEN ADJUST FOR QUADRANT
A185                    SSIN
A185 20 6F A3                   JSR     SINMOD          ;TAKE ANGLE MOD 2*PI, 360
                        ;
                        ; FR0=FR0/(PI/2) OR FR0=FR0/90
A188 20 A0 A3                   JSR     PIOVL           ;LOAD X & Y REGS TO GET PI/2 OR 90
A18B 20 0F A3                   JSR     LD1DIV          FR0=FR0/FR1
                                                        NOW HAVE 0-4 (NOT NECESSARILY INTEGER)
                        ; IF FR0 NOW FRACTION, IT IS QUADRANT 0
                        ; ELSE, GET INTEGER OF FR0 LSD
A18E A9 00                      LDA     #0
A190 8D 80 04                   STA     QUADFLG         ASSUME QUADRANT 0
A193 A5 D4                      LDA     FR0             GET EXPONENT
A195 C9 40                      CMP     #$40            SUBTRACT 64 EXCESS
A197 90 19                      BCC     SINF3           GO IF QUADRANT 0
                        ;
A199 A5 D5                      LDA     FR0+1           ;SHOULD BE 0, 1, 2, OR 3
A19B 8D 80 04                   STA     QUADFLG         NOW HAVE QUADRANT (0,1,2, OR 3)
                        ;
A19E 20 B6 DD                   JSR     FMOVE           ;FR1 <- FR0
A1A1 20 31 A3                   JSR     STRUNC          ;TRUNCATE FR0
A1A4 20 07 A3                   JSR     SFSUB           ;FR0 <- TRUNC(FR0)-FR0
A1A7 20 28 A3                   JSR     SCHGSG          ; CHANGE SIGN -- FRACTIONAL PART (FR0) = FR0 - TRUNC (FR0)
                        ;
                        ; IF ODD QUADRANT, SET FR0=1-FR0 (90 DEGREE INVERT)
A1AA 4E 80 04                   LSR     QUADFLG         IS IT ODD QUADRANT?
A1AD 90 03                      BCC     SINF3           NO
A1AF 20 FD A2                   JSR     ONESUB          ;FR0 <- 1-FR0
                        ;
                        ; SAVE ARG FOR LATER
A1B2                    SINF3
A1B2 A2 E6                      LDX     #FPSCR
A1B4 A0 05                      LDY     #FPSCR/256
A1B6 20 A7 DD                   JSR     FSTOR           ;FPSCR <- FR0
                        ;
                        ; NOW COMPUTE SINE
                        ; THIS CODE TAKEN FROM BASIC 5.9 LINES 6760-6770
A1B9 20 EE A2                   JSR     SSQUAR          FR0=X**2
A1BC A9 06                      LDA     #NSCF
A1BE A2 AF                      LDX     #SCOEF
A1C0 A0 A3                      LDY     #SCOEF/256
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
A1BC A9 06              LDA     #NSCF
A1BE A2 AF              LDX     #SCOEF
A1C0 A0 A3              LDY     #SCOEF/256
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
A1C2 20 40 DD           JSR     PLYEVL          EVALUATE P(X**2)
A1C5 A2 E6              LDX     #FPSCR
A1C7 A0 05              LDY     #FPSCR/256
A1C9 20 E8 A2           JSR     LD1MUL          FR0=SIN(X)=X*P(X**2)
                        ;
                        ; IF LOWER QUADRANT (2 OR 3) THEN FR0=-(FR0)
A1CC 4E 80 04           LSR     QUADFLG         IS IT LOWER QUAD?
A1CF 90 03              BCC     SINF4           NO
A1D1 20 28 A3           JSR     SCHGSG          ;YES

A1D4                    SINF4
                        ;
                        ; IF ABS(FR0) >= 1 THEN SET TO 1
A1D4 A5 D4              LDA     FR0
A1D6 29 7F              AND     #$7F            WITHOUT SIGN BIT
A1D8 C9 40              CMP     #$40            COMPARE $40
A1DA 90 07              BCC     SINFIN
A1DC A9 00              LDA     #0
A1DE 85 D8              STA     FR0+4           ;PERFORM PSEUDO INT(FR0)  (CLEAR LAST 2 BYTES)
A1E0 85 D9              STA     FR0+5
A1E2 18         SINFN2  CLC                     ;NO ERROR
A1E3 60         SINFIN  RTS
```

```
                        ;
                        ;       FRO <- ARC TANGENT (FRO)
                        ;
                        ;       FROM SHEPARDSON ATARI BASIC 5.9 4-5-79 (MODIFIED)
                        ;       SAME ACCURACY AS SHEP VERSION -- USES FEWER BYTES
                        ;
A1E4                    SATAN
A1E4 A9 00                      LDA     #0
A1E6 85 F0                      STA     FCHRFLG         ;SIGN FLAG OFF
A1E8 85 F1                      STA     DIGRT           ;AND TRANSFORM FLAG
A1EA A5 D4                      LDA     FRO
A1EC AA                         TAX
A1ED 29 7F                      AND     #$7F
A1EF C9 40                      CMP     #$40            ;CHECK X VS 1.0
A1F1 30 10                      BMI     ATAN1           ;X<1 - USE SERIES DIRECTLY
A1F3 85 D4                      STA     FRO             ;FORCE PLUS
A1F5 8A                         TXA                     ;OLD FRO WITH SIGN
A1F6 29 80                      AND     #$80
A1F8 85 F0                      STA     FCHRFLG         ;REMEMBER SIGN
A1FA E6 F1                      INC     DIGRT
A1FC A2 EA                      LDX     #FP9S
A1FE A0 DF                      LDY     #FP9S/$100
A200 20 95 DE                   JSR     XFORM           ;CHANGE ARG TO (X-1)/(X+1)
A203                    ATAN1
                                                        ARCTAN(X), -1<X<1 BY SERIES APPROX
A203 A2 E6                      LDX     #FPSCR          ;CAN'T USE FTEMP BECAUSE SATAN IS CALLED BY OTHER ROUTINES WHICH USE IT
A205 A0 05                      LDY     #FPSCR/256
A207 20 A7 DD                   JSR     FSTOR
A20A 20 EE A2                   JSR     SSQUAR          ;X*X -> FRO
A20D A9 0B                      LDA     #NATCF
A20F A2 AE                      LDX     #ATCOEF
A211 A0 DF                      LDY     #ATCOEF/256
A213 20 40 DD                   JSR     PLYEVL          ;P(X*X)
A216 B0 26                      BCS     ATNOUT          ;ERROR
A218 A2 E6                      LDX     #FPSCR
A21A A0 05                      LDY     #FPSCR/256
A21C 20 E8 A2                   JSR     LD1MUL          ;X*P(X*X)
A21F A5 F1                      LDA     DIGRT           ;WAS ARG XFORMED
A221 F0 10                      BEQ     ATAN2           ;NO.
A223 A2 F0                      LDX     #PIOV4          ;YES-ADD ARCTAN(1) = PI/4
A225 A0 DF                      LDY     #PIOV4/256
A227 20 98 DD                   JSR     FLD1R
A22A 20 66 DA                   JSR     FADD
A22D A5 F0                      LDA     FCHRFLG         ;GET ORG SIGN
A22F 05 D4                      ORA     FRO
A231 85 D4                      STA     FRO             ;ATAN(-X) = -ATAN(X)
A233                    ATAN2
A233 A5 FB                      LDA     RADFLG          ;RAD OR DEG
A235 F0 07                      BEQ     ATNOUT          ;RAD - FINI
A237 A2 D9                      LDX     #PIOV18
A239 A0 A3                      LDY     #PIOV18/256     ;DIVIDE BY PI/180 TO CONVERT TO DEGREES
A23B 20 0F A3                   JSR     LD1DIV
A23E                    ATNOUT
A23E 60                         RTS
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
                        ;
                        ;       FRO <- SQUARE ROOT (FRO)
                        ;
                        ;       FROM SHEPARDSON ATARI BASIC 5.9 4-5-79 (MODIFIED)
                        ;       SAME ACCURACY AS SHEP VERSION -- USES FEWER BYTES
                        ;
                        ;       USES NEWTON-RAPHSON ITERATION
                        ;       F(Y) = Y*Y - X
                        ;       FPRIME(Y) = 2*Y
                        ;       Y[I+1] = Y[I] - F(Y[I]) / FPRIME(Y[I]) = Y[I] + .5*((X/Y[I])-Y[I])
                        ;
                        ;       ERROR EXIT
                        ;
A23F                    SQRERR
A23F 38                         SEC
A240 60                         RTS
                        ;
                        ;       ENTRY POINT
                        ;
A241                    SSQRT                           ;X<-SQRT(X)
A241 A2 E0                      LDX     #FR1
A243 20 46 DA                   JSR     ZF1             ;FR1 <- ALL 0'S
A246 A2 00                      LDX     #0
A248 86 F1                      STX     DIGRT
A24A E8                         INX                     ;1
A24B 86 E1                      STX     FR1+1
A24D A5 D4                      LDA     FRO
A24F 30 EE                      BMI     SQRERR          ;ERROR IF <0
A251 C9 3F                      CMP     #$3F
A253 F0 09                      BEQ     FSQR            ;X IN RANGE OF APPROX - GO DO IT TO IT
A255 AA                         TAX
A256 E8                         INX
A257 86 E0                      STX     FR1             ;MANTISSSA = 1
A259 86 F1                      STX     DIGRT           ;NOT IN RANGE - TRANSFORM
A25B 20 28 DB                   JSR     FDIV            ;X/100**N
A25E                    FSQR                            ;SQR(X) 0.1<=X<1
A25E A9 06                      LDA     #6
A260 85 EF                      STA     ESIGN
A262 A2 E6                      LDX     #FSCR
A264 A0 05                      LDY     #FSCR/256
A266 20 A7 DD                   JSR     FSTOR           ;STASH X IN FSCR
A269 A9 02                      LDA     #2
A26B 20 FF A2                   JSR     INTSUB          ;2-X
A26E A2 E6                      LDX     #FSCR
A270 A0 05                      LDY     #FSCR/256
A272 20 E8 A2                   JSR     LD1MUL          ;X*(2-X) : 1ST APPROX
A275                    SQRLP
A275 A2 EC                      LDX     #FSCR1
A277 A0 05                      LDY     #FSCR1/256
A279 20 A7 DD                   JSR     FSTOR           ;Y->FSCR1
A27C 20 B6 DD                   JSR     FMOVE           ;Y->FR1
A27F A2 E6                      LDX     #FSCR
A281 A0 05                      LDY     #FSCR/256
A283 20 89 DD                   JSR     FLDOR
A286 20 28 DB                   JSR     FDIV            ;X/Y
A289 A2 EC                      LDX     #FSCR1
A28B A0 05                      LDY     #FSCR1/256
A28D 20 98 DD                   JSR     FLD1R
A290 20 60 DA                   JSR     FSUB            ;(X/Y)-Y
A293 A2 6C                      LDX     #FHALF
```

```
A295 A0 DF                      LDY     #FHALF/256
A297 20 E8 A2                   JSR     LD1MUL          ;.5*((X/Y)-Y)=DELTAY
A29A A5 D4                      LDA     FR0             ;DELTA 0
A29C F0 0E                      BEQ     SQRDON
A29E A2 EC                      LDX     #FSCR1
A2A0 A0 05                      LDY     #FSCR1/256
A2A2 20 98 DD                   JSR     FLD1R
A2A5 20 66 DA                   JSR     FADD            ;Y=Y+DELTA Y
A2A8 C6 EF                      DEC     ESIGN           ;COUNT & LOOP
A2AA 10 C9                      BPL     SQRLP
A2AC                    SQRDON
A2AC A2 EC                      LDX     #FSCR1          ;DELTA = 0 - GET Y BACK
A2AE A0 05                      LDY     #FSCR1/256
A2B0 20 89 DD                   JSR     FLDOR

A2B3 A2 E0                      LDX     #FR1            WAS ARG TRANSFORMED?
A2B5 20 46 DA                   JSR     ZF1             ;FR1 <- ALL 0'S AGAIN
A2B8 A5 F1                      LDA     DIGRT
A2BA F0 16                      BEQ     SABSVA          ;NO FINI
A2BC 38                         SEC
A2BD E9 40                      SBC     #$40

A2BF 4A                         LSR     A               ;YES - TRANSFORM RESULT TO MATCH
A2C0 08                         PHP                     ;DIVIDE EXP BY 2
A2C1 18                         CLC                     ;SAVE CARRY (LSB OF DIGRT)
A2C2 69 40                      ADC     #$40
A2C4 85 E0                      STA     FR1
A2C6 A9 01                      LDA     #1
A2C8 28                         PLP                     ;MANTISSA = 1
A2C9 90 02                      BCC     SQR2            ;RELOAD CARRY (LSBIT OF DIGRT)
A2CB A9 10                      LDA     #$10            ;WAS EXP ODD OR EVEN
A2CD                    SQR2                            ;ODD - MANT = 10
A2CD 85 E1                      STA     FR1+1
A2CF 20 DB DA                   JSR     FMUL            ;SQR(X) = SQR(X/100**N) * (10**N)
A2D2                    SABSVA                          ;FR0 <- ABSVAL(FR0)  A<-FR0
A2D2 A5 D4                      LDA     FR0
A2D4 29 7F                      AND     #$7F
A2D6 85 D4                      STA     FR0
A2D8                    SABRTN
A2D8 60                         RTS
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
                        ;
                        ;       THE FOLLOWING ROUTINES ARE CALLED BY THE PREVIOUS ROUTINES
                        ;       IN GENERAL, THEY DO A 2-LEVEL RETURN WITH CARRY SET IF AN
                        ;       ERROR OCCURS, THUS BYPASSING THE REMAINDER OF THE CALLING ROUTINE
                        ;
A2D9                    FSTIR                           ;LIKE FSTOR EXCEPT USES FR1
A2D9 86 FC                      STX     FLPTR
A2DB 84 FD                      STY     FLPTR+1
A2DD A0 05                      LDY     #5
A2DF                    FSLOP
A2DF B9 E0 00                   LDA     FR1,Y
A2E2 91 FC                      STA     (FLPTR),Y
A2E4 88                         DEY
A2E5 10 F8                      BPL     FSLOP
A2E7 60                         RTS

A2E8                    LD1MUL                          ;FR0 <- FR0 * DATA CONSTANT (ADDR IN X & Y)
A2E8 20 98 DD                   JSR     FLD1R
A2EB 4C F1 A2                   JMP     SFMUL

A2EE                    SSQUAR
A2EE 20 B6 DD                   JSR     FMOVE           ;FR0 <- FR0 * FR0

A2F1                    SFMUL                           ;FR0 <- FR0 * FR1
A2F1 20 DB DA                   JSR     FMUL
A2F4 B0 16                      BCS     CRYSND
A2F6 60                         RTS

A2F7                    SFADD                           ;FR0 <- FR0 + FR1
A2F7 20 66 DA                   JSR     FADD
A2FA B0 10                      BCS     CRYSND
A2FC 60                         RTS

A2FD A9 01              ONESUB  LDA     #1              ;FR0 <- 1-FR0

A2FF                    INTSUB                          ;FR0 <- A - FR0
A2FF 48                         PHA
A300 20 B6 DD                   JSR     FMOVE
A303 68                         PLA
A304 20 53 A3                   JSR     PSETO           ;A MUST BE FROM 0-9 OR BCD

A307                    SFSUB                           ;FR0 <- FR0 - FR1
A307 20 60 DA                   JSR     FSUB
A30A                    CRYCHK                          ;CHECK CARRY TO SEE IF THERE IS AN ERROR
A30A 90 02                      BCC     RETURN          ;RETURN IF CARRY CLEAR
A30C                    CRYSND                          ;DO A 2-LEVEL RETURN IF ERROR
A30C 68                         PLA
A30D 68                         PLA
A30E 60                 RETURN  RTS

A30F                    LD1DIV                          ;FR0 <- FR0 / (X,Y)
A30F 20 98 DD                   JSR     FLD1R
A312                    SFDIV                           ;FR0 <- FR0 / FR1
A312 20 28 D8                   JSR     FDIV
A315 B0 F5                      BCS     CRYSND
A317 60                         RTS
```

```
A318                    XINT60                          ;FRO <- INT(FRO)
A318 A5 D4                      LDA     FRO
A31A 48                         PHA
A31B 20 31 A3                   JSR     STRUNC          ;FRO <- TRUNC(FRO), RETURN EQ IF ALREADY INT
A31E F0 2F                      BEQ     INTRT3          ;INTEGER   POP AND RETURN
A320 68                         PLA                     ;RELOAD OLD FRO WITH SIGN
A321 10 0D                      BPL     INTRT2          ;POSITIVE

                                                        WAS NEGATIVE NON-INTEGER

A323                    SUBONE                          ;FRO <- FRO-1
A323 A9 01                      LDA     #1


A325                    SUBINT                          ;FRO <- FRO - A
A325 20 FF A2                   JSR     INTSUB          ;FRO <- A-FRO
A328                    SCHQSQ
A328 A5 D4                      LDA     FRO             ;FRO<- -FRO  A<-FRO  SET EQ/NE
A32A F0 04                      BEQ     SCH10
A32C 49 80                      EOR     #$80
A32E 85 D4                      STA     FRO
A330                    SCH10
A330 60                         RTS
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

```
                                                GREATEST INT <= FRO

                                PART OF INT ROUTINE FROM SHEP ATARI BASIC BOD5-BOEE
                                DOES NOT AFFECT FR1?

A331                    STRUNC                  ;TRUNCATE FRO
                                                RETURN A=0 AND EQ IF FRO WAS ALREADY AN INTEGER

A331 A5 D4                      LDA     FRO     ;GET EXPONENT
A333 29 7F                      AND     #$7F    ;AND OUT SIGN BIT
A335 38                         SEC
A336 E9 3F                      SBC     #$3F    ;GET LOCATION OF 1ST FRACTION BYTE
A338 10 02                      BPL     XINT1   ;IF >= 0 THEN BRANCH
A33A A9 00                      LDA     #0      ;ELSE SET =0
A33C                    XINT1
A33C AA                         TAX             ;PUT IN X AS INDEX INTO FROM
A33D A9 00                      LDA     #0      ;SET ACCUM TO ZERO FOR ORING
A33F A8                         TAY             ;ZERO Y
A340                    INT2
A340 E0 05                      CPX     #FPREC-1 ;IS D.P. LOC >= 5?
A342 B0 07                      BCS     INTRTN  ;IF YES, LOOP DONE
A344 15 D5                      ORA     FRO+1,X ;OR IN THE BYTE OF MANTISSA
A346 94 D5                      STY     FRO+1,X ;ZERO BYTE
A348 E8                         INX             ;POINT TO NEXT BYTE
A349 D0 F5                      BNE     INT2    ;JMP
A34B                    INTRTN
A34B 48                         PHA             ;SAVE OR OF ALL FRACTIONAL BYTES
A34C 20 00 DC                   JSR     NORM    ;NORMALIZE
A34F                    INTRT3
A34F 68                         PLA             ;RELOAD
A350 60                 INTRT2  RTS


A351                    PCLRO                   ;CLEAR FRO
                                                RETURN WITH CARRY CLEAR (CC)
A351 A9 00                      LDA     #0
A353                    PSETO                   ;SET FRO TO INTEGER PASSED IN A (MUST BE BCD OR <10)
                                                RETURN WITH CARRY CLEAR (CC)
A353 48                         PHA
A354 20 44 DA                   JSR     ZFRO    ;FRO <- 0
A357 68                         PLA
A358 F0 06                      BEQ     CLRTN   ;0 => ALL 0'S
A35A 85 D5                      STA     FRO+1
A35C A9 40                      LDA     #$40    ;SET EXPONENT
A35E 85 D4                      STA     FRO
A360                    CLRTN
A360 18                         CLC
A361 60                         RTS
```

```
                        ;       SINE ROUTINES
A362                    SINLD
A362 A2 DF                      LDX     #PI2            ;LOAD 2*PI
A364 A0 A3                      LDY     #PI2/256
A366 A5 F8                      LDA     RADFLG
A368 F0 08                      BEQ     SNMOD3
A36A A2 E5                      LDX     #C360           ;DEGREES => LOAD 360
A36C A0 A3                      LDY     #C360/256
A36E                    SNMOD3
A36E 60                         RTS

A36F                    SINMOD                          ;FIND ANGLE MOD 2*PI OR 360 DEPENDING ON RADFLG
A36F A5 D4                      LDA     FR0
A371 29 7F                      AND     #$7F
A373 C9 45                      CMP     #$45
A375 B0 98                      BCS     CRYSND          ;OUT OF RANGE -- 2-LEVEL RETURN
A377 A2 E6                      LDX     #FPSCR          ;SAVE IN TEMP SCRATCH REG
A379 A0 05                      LDY     #FPSCR/256
A37B 20 A7 DD                   JSR     FSTOR
A37E 20 62 A3                   JSR     SINLD           ;LOAD 2*PI OR 360
A381 20 98 DD                   JSR     FLD1R
A384 20 12 A3                   JSR     SFDIV           ;ANGLE/360
A387 20 18 A3                   JSR     SINTEG          ;INT(ANGLE/360)
A38A 20 62 A3                   JSR     SINLD           ;LOAD 2*PI OR 360
A38D 20 98 DD                   JSR     FLD1R
A390 20 F1 A2                   JSR     SFMUL           ;INT(ANGLE/360)*360
A393 20 B6 DD                   JSR     FMOVE
A396 A2 E6                      LDX     #FPSCR          ;RELOAD ANGLE
A398 A0 05                      LDY     #FPSCR/256
A39A 20 89 DD                   JSR     FLDOR
A39D 4C 07 A3                   JMP     SFSUB           ;ANGLE - INT(ANGLE/360)*360

A3A0                    PIOVL           ;LOAD X & Y REGS IN PREPARATION FOR LOADING REG 0 OR 1 WITH PI/2, 90 OR 100(IF GRAD)
A3A0 A9 CD                      LDA     #RADPI2
A3A2 18                         CLC
A3A3 65 FB                      ADC     RADFLG
A3A5 AA                         TAX
A3A6 A0 A3                      LDY     #RADPI2/256
A3A8 60                         RTS
```

```
COLLEEN FLOATING POINT ROUTINES BY C SHAW

                                        DATA

A3A9 40 01 00       ONE     BYTE    $40,$01,0,0,0,0         ;1
A3AC 00 00 00
A3AF                SCOEF
A3AF BD 03 55                BYTE    $BD,$03,$55,$14,$99,$39 ;- 00000355149939
A3B2 14 99 39
A3B5 3E 01 60                BYTE    $3E,$01,$60,$44,$27,$52 ;0 000160442752
A3B8 44 27 52
A3BB BE 46 B1                BYTE    $BE,$46,$B1,$75,$43,$55 ;- 004681754355
A3BE 75 43 55
A3C1 3F 07 96                BYTE    $3F,$07,$96,$92,$62,$39 ;0 0796926239
A3C4 92 62 39
A3C7 BF 64 59                BYTE    $BF,$6A,$59,$64,$08,$67 ;- 6459640867
A3CA 64 08 67
A3CD 40 01 57       RADFIX   BYTE    $40,$01,$57,$07,$96,$32 ;PI/2 = 1 570796327  PART OF SCOEF
A3D0 07 96 32
A3D3 40 90 00                BYTE    $80,$90,0,0,0,0         ;90 (DEGREES)
A3D6 00 00 00
A3D9 3F 01 7A       PIOVIS   BYTE    $3F,$01,$74,$53,$29,$25 ;PI/180 =  0174532925 DEG->RAD
A3DC 53 29 25
A3DF 40 06 28       PI2      BYTE    $40,$06,$28,$31,$85,$31 ;2*PI = 6 28318531
A3E2 31 85 31
A3E5 41 03 60       CON0     BYTE    $41,$03,$60,0,0,0       ;360
A3E8 00 00 00

                            **BFFA
BFFA 00 A0                  WORD    START           ;CARTRIDGE START INFO
BFFC 00 04                  BYTE    0,4             ;COLD/WARM START ADDRESS
BFFE CD A0                  WORD    INIT            ;RUN CARTRIDGE
                            END                     ;POWER UP START VECTOR
```

COLLEEN FLOATING POINT ROUTINES BY C SHAW

SYMBOL TABLE

AFP	D800	ATAN1	A202	ATAN2	A233	ATCOEF	DFAE
ATNOUT	A23E	C360	A2E5	CIOV	E456	CIX	00F2
CLRTN	A360	CONTIN	A05B	CR	009B	CRYCHK	A30A
CRYSND	A30C	DIGRT	00F1	EEXP	00ED	ERRMSG	A079
ESIGN	00EF	EXP	DDC0	EXP1	DE03	EXP10	DDCC
EXP11	DE12	FADD	DA66	FASC	D8E6	FASC2	D905
FCHKFL	00F0	FDIV	DB2B	FHALF	DF6C	FLDOR	DD89
FLD1R	DD98	FLPTR	00FC	FMOVE	DDB6	FMUL	DADB
FPPS	DFEA	FPI	D9D2	FPREC	0006	FPSCR	05E6
FPSCR1	05EC	FPTR2	00FE	FR0	00D4	FR1	00E0
FR2	00E6	FRE	00DA	FRX	00EC	FSCR	05E6
FSCR1	05EC	FSLDP	92DF	FSQR	A25E	FSTOR	DDA7
FSTR	A2D9	FSUB	DA60	FTEMP	0482	GETNUM	A04C
GETREC	0005	ICBAL	0344	ICBLL	0348	ICCOM	0342
IFP	D9AA	INBUFF	00F3	INIT	A0CD	INT2	A340
INTFLG	0481	INTLBF	DA51	INTRT2	A350	INTRT3	A34F
INTRTN	A34B	INTSUB	A2FF	LBPR1	057E	LBPR2	057F
LBUFF	0580	LD1DIV	A30F	LD1MUL	A2E8	LOG	DECD
LOG10	DED1	LOG10E	DE89	LOGCHK	A15E	LOGCLP	A167
MLOOP	A01D	NATCF	000B	NOERR	A018	NORM	DC00
NSCF	0006	NSIGN	00EE	ONE	A3A9	ONESUB	A2FD
PCLR0	A351	PERR2	A150	PERROR	A14F	PI2	A3DF
PIOV18	A3D9	PIOV4	DFF0	PIOVL	A3A0	PLYARG	05E0
PLYEVL	DD40	PRTN	A151	PSET0	A353	PULRTN	A176
PUTREC	0009	QUADFL	0480	RADFLG	00FB	RADPI2	A3CD
RETURN	A30E	RTURN2	A178	SABRTN	A2D8	SABSVA	A2D2
SATAN	A1E4	SCH10	A330	SCHGSG	A328	SCOEF	A3AF
SCOS	A179	SEXP05	A08D	SEXP10	A0C9	SEXPE	A07F
SEXPTE	A086	SFADD	A2F7	SFDIV	A312	SFERR3	A0CC
SFMUL	A2F1	SFSUB	A307	SINF3	A1B2	SINF4	A1D4
SINFIN	A1E3	SINFN2	A1E2	SINLD	A362	SINMOD	A36F
SINTEG	A318	SLN	A152	SLOGTE	A158	SNMOD3	A36E
SPOW10	A0DC	SPOW20	A0DF	SPOW50	A109	SPOW80	A140
SPOWER	A0CE	SQR2	A2CD	SQRDON	A2AC	SQRERR	A23F
SQRLP	A275	SROU10	A13A	SSIN	A185	SSQRT	A241
SSQUAR	A2EE	START	A000	STRUNC	A331	SUBINT	A325
SUBONE	A323	XCVFR0	DC70	XEFORM	D920	XEFRM2	D928
XFORM	DE95	XINT1	A33C	ZF1	DA46	ZFR0	DA44
ZTEMP1	00F5	ZTEMP3	00F9	ZTEMP4	00F7		